Simulation advances using the ESL Simulation Language and the Virtual Test Bed

John G Pearce
ISIM International Simulation Limited
161 Claremont Road
Salford, M6 8PA, UK
mailto:johnpearce@isimsimulation.com

Keywords: Virtual Test Bed, VTB, European Simulation Language, ESL, Continuous System Simulation

Abstract

The Virtual Test Bed (VTB) is a software environment for constructing simulations of large scale multidisciplinary dynamic systems. The VTB provides a common platform in which component models developed by different teams using different tools can be merged. ESL is one such simulation tool which was developed for the European Space Agency for robust dynamic simulation of complex systems. This paper describes progress in the development of an interface between the ESL simulation tool and the professional release of the Virtual Test Bed (VTB Pro). The chosen technique for the interface to VTB Pro is through the use of a bespoke ESL Importer program which takes .NET CLR assemblies generated from ESL models; allows a user to create a graphical icon to represent the model in a VTB diagram; and adds the model to the VTB component database. This facility allows the advantages of ESL models (differential equation representation, accurate non-linearity handling and proven robustness) to be made available through the graphical environment of VTB Pro. In order to illustrate the use of the new VTB facility, the implementation of a simulation of a multi-legged mobile robot (Genghis – originally developed for the European Space Agency to demonstrate ESL's distributed simulation capability) is described.

1. INTRODUCTION

The Virtual Test Bed (VTB) is a software environment for developing simulations of large scale multidisciplinary dynamic systems. It allows alternative designs to be analysed and tested before being committed to manufacture. The main application that is driving the development of the VTB is a need to model advanced power systems for navy ships. In such systems there are many different energy generation and storage devices including nuclear, fuel cells and gas turbines. The distribution networks are also of unconventional design having dc power busses and high numbers of interconnections that can be rapidly reconfigured. Constructing complete coherent simulations of such large scale systems, involving widely differing technologies poses a serious challenge. Each discipline group will use their preferred simulation tool to model their part of the whole system. The VTB aims to satisfy this challenge by providing a common platform in which component models developed by different teams using different tools can be merged. A companion program, Eye-Sys, may be used to link three dimensional visualizations to a VTB simulation.

ESL [Crosbie et al 1981, Hay et al 1994, Pearce and Crosbie 2000] is an advanced high-level simulation language for modelling large-scale systems from a variety of disciplines. ESL is an acronym of the *European Simulation Language* (originally European Space Agency Simulation Language) and comprises two components: the language itself and a graphical user interface - the Integrated Simulation Environment (ISE). ESL is a continuous system simulation language and is used for modelling dynamic systems which are usually described by ordinary and partial differential equations. ISE provides the environment from which all stages of the simulation process can be managed. The software was developed mainly through a series of contracts with ESTEC - the European Space Research and Technology Centre - part of ESA with additional support from various industrial simulation consultancy activities.

In this paper ESL is used as an example to illustrate the process of integrating models generated using other tools with the professional release of VTB (VTB Pro). The work described is a continuation of an earlier project involving integration with VTB 2003 [Pearce 2007].

2. ESL SEGMENTS

A feature of ESL that lends itself to integration with other software is its *segment* structures. Segments were originally included in ESL as a means of providing a parallel processing capability. The idea was that a large simulation could be broken down into self-contained segments that could be executed in parallel on different processors or networked computers. Communication takes place between segments at pre-determined communication points through a TCP IP protocol. There are three types of segment in ESL:

- *Remote segments* – these can be assigned to different processors for truly parallel operation.
- *Emulated segments* – these allow parallel operation to be emulated on a single computer – useful for testing parallel segments before assignment to separate processors and for implementing multi-rate simulations.
- *Embedded segments* – used where an ESL model is to be integrated with another application.

An ESL embedded segment is a particular way of expressing a dynamic model in which all interface variables appear in special structures call *Packages*. Figure 1 is an example of an embedded segment for a simple linear model of a dc motor. Inputs to the model appear in the package Esl_inp; State outputs appear in the package Esl_state and algebraic ouputs in package Esl_out. The package Esl_par contains parameters which should be accessible to the user and Esl_view contains viewables, i.e. any variables that may be plotted are used to drive visualizations. The dynamic model itself is defined in the *Segment* structure.

Using an ESL utility, *eslgen*, an embedded segment may be compiled into a dynamic link library (DLL), a COM server or a .NET CLR assembly. The .NET CLR route is used for the VTB Pro integration. The .NET CLR assembly that is generated from the embedded segment provides access to the package variables and a number of functions required to run the model (see Table 1). The assembly also provides access to specific simulation parameters (table 2) and those variables defined in the embedded segment packages.

The embedded segment features described above are completely general and can be accessed from any .NET compatible program. A further stage of processing is required to make the ESL embedded segment accessible from the VTB.

Table 1 ESL .NET CLR assembly functions

Name	***Meaning***
ExStrt	Prepare embedded code for use - must only be used once at program start.
ExInit	Initialise embedded segment for a single simulation run.
ExSim	Advance Simulation by one time-frame (specified by the simulation parameter CINT).
ExPrestep	Evaluate algebraic outputs without advancing the simulation
ExFin	Close down simulation - must only be used once at program termination.

```
EMBEDDED
Package Esl_inp;
  Real: va, tl;
End Esl_inp;
--
Package Esl_state;
  Real: ia, Wa;
End Esl_state;
--
Package Esl_out;
  Real: v_error;
End Esl_out;
--
Package Esl_par;
  Parameter Real:Kt/0.0275/,
    Kb/0.04/, Ra/9.0/,
    La/4.065e-03/, Ja/1.71e-06/,
    Ba/1.5e-04/;
End Esl_par;
--
Package Esl_view;
  Real: v_back, t_motor, t_avail;
End Esl_view;
--
Segment dc_motor;
  Use Esl_inp; Use Esl_state;
  Use Esl_out; Use Esl_par;
  Use Esl_view;
  Real: i, w, ve, vb, tm, ta;
  Dynamic
    ve:= va-vb;
    i:=Transfer(1/(La*s + Ra))*ve;
    tm:= Kt*i; ta:= tm-tl;
    w:=  Transfer(1/(Ja*s+Ba))*ta;
    vb:= Kb*w;
  Communication
    ia := i; wa := w;
    v_back  := vb; v_error := ve;
    t_motor := tm; t_avail := ta;
End dc_motor;
```

Figure 1 dc motor embedded segment

3. INTEGRATION WITH THE VTB

3.1. ESL Importer

In order to be accessed from the VTB the ESL .NET assembly must appear in VTB as an *entity* (entities are the basic VTB simulation elements). This is achieved through a wrapper (ESL Component) that is automatically generated using a VTB *ESL Importer* utility. The wrapper accesses the specific ESL .NET assembly functions (Table 1) and data (as specified in the packages) while providing a generic interface with the VTB. This arrangement is shown in Figure 2.

The ESL Importer (which is based on the standard VTB entity builder) enables a user to:

- Choose a default icon for the ESL entity or design a custom icon.
- Determine which of the inputs and outputs specified in the embedded segment packages are to appear as ports on the icon.
- Change default values of model parameters and specify which are to be accessible from VTB at run-time.
- Set the entity name and add comments and units for all ports, parameters and viewables.

Figure 3 is a screen dump showing the appearance of the dc motor example opened in the ESL Importer.

Table 2 ESL simulation parameters

Parameter	***Default value***	***Meaning***
T		Current value of time**.**
Tstart	0.0	Initial value of time (T) at start of run.
Tfin	10.0	Final value of T at end-of-run.
Cint	1.0	Communication interval – must be same as VTB.
Diserr	0.0001	Discontinuity detection error tolerance.
Interr	0.001	Integration error tolerance.
Algo	1 (RK5)	Integration algorithm (8 algorithms are available).
Nstep	1	Minimum number of integration steps in CINT.

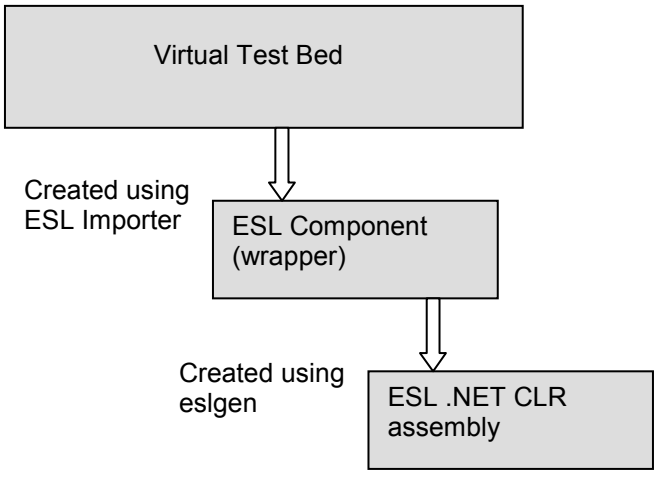

Figure 2 VTB – ESL interface

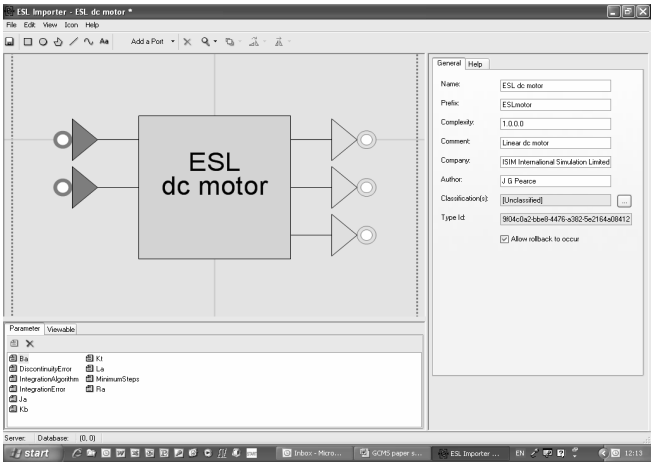

Figure 3 ESL Importer

Once ESL entities are created they are loaded in the VTB component database from which they are available for inclusion in any simulation schematic.

3.2. Execution Considerations

The following sequence of operations takes place when a VTB simulation containing ESL entities is executed:

- When the entity is first loaded, the function ExStrt is called to initialize the embedded segment code.
- Prior to any simulation run, input values are passed to the entity and the function ExInit is called to initialize the embedded segment. At this point all entity outputs will have been correctly initialized.
- In order to advance the embedded segment solution by one VTB time-step, the entity inputs are updated and the function ExSim is called. During this time step *all entity inputs are held constant*. At the end of the step all state entity outputs will have been updated.
- Before commencing a new simulation step, any algebraic entity outputs are evaluated using the function ExPrestep as part of a VTB pre signal-step sequence which resolves all algebraic relationships within the simulation.

4. ESL – VTB DEMONSTARTION APPLICATION

In order to evaluate the ESL-VTB interface more fully and provide a measure of performance, a more substantial application was required. To this end, an existing ESL simulation of a six legged walking robot – "Genghis" [Pearce 1993, Hay et al 1994] – was modified to be run under the VTB.

The simulation was originally undertaken under an ESA contract to demonstrate ESL's *distributed* and *real-time* simulation capability. The simulation comprised a kinematic model of the autonomous robot (which included edge avoidance, mutual avoidance and tracking algorithms);

an interactive graphical control panel, from which the robot's speed and direction could be controlled; and a main module which coordinated the other components and generated graphical data for a 3D visualization. For the ESA contract the simulation ran on Sun workstations – two instances of the robot model plus two corresponding control panels were supported, each (in principle) running on different networked computers. A separate Silicon Graphics computer graphically rendered the model using visualization software written by EADS-CASA of Spain. The robots had the capability of simultaneously negotiating an arbitrarily defined uneven terrain while tracking a moving target, or each other. Figure 4 shows the original architecture and Figure 5 the appearance of the robot and control panel as rendered by the Silicon Graphics computer.

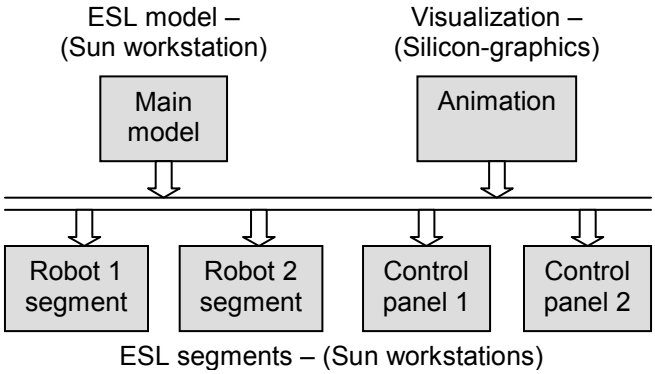

Figure 4 Original Genghis architecture

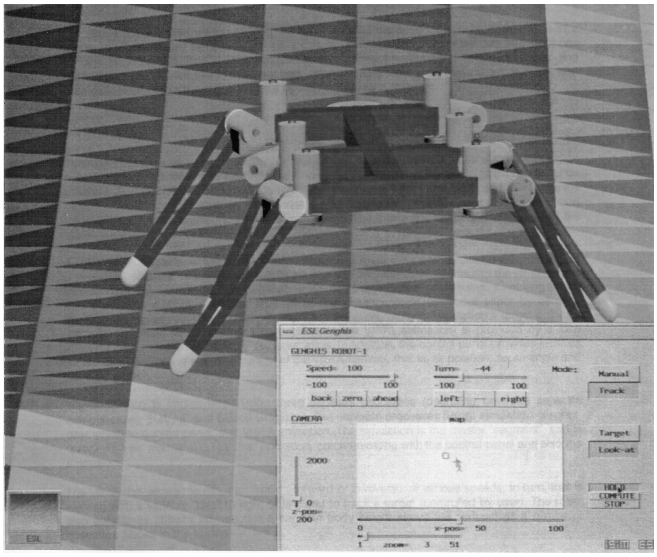

Figure 5 Original Genghis on Silicon Graphics display

In order to demonstrate multiple ESL entities interacting with each other in a VTB simulation, embedded segments were created for the Main model, the Robot and the Control panel. These segments defined the I/O ports, parameters and viewables in their respective packages, as described in section 2. ESL entities were generated from the embedded segments and connected in a VTB schematic. Figures 6 shows the new VTB architecture and Figure 7 is a screen-dump of the actual VTB schematic. The objective was to have fairly complex entities with multiple connections – clearly achieved in this application.

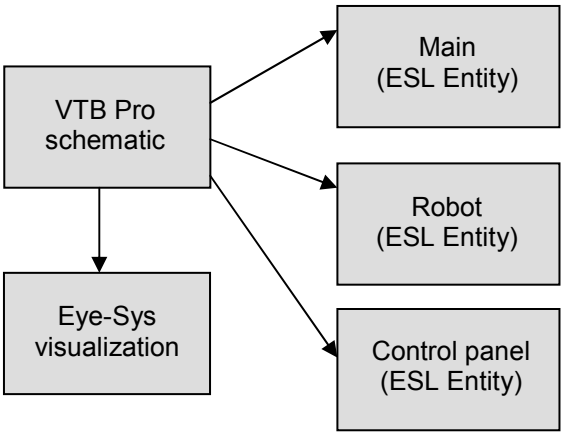

Figure 6 VTB Genghis architecture

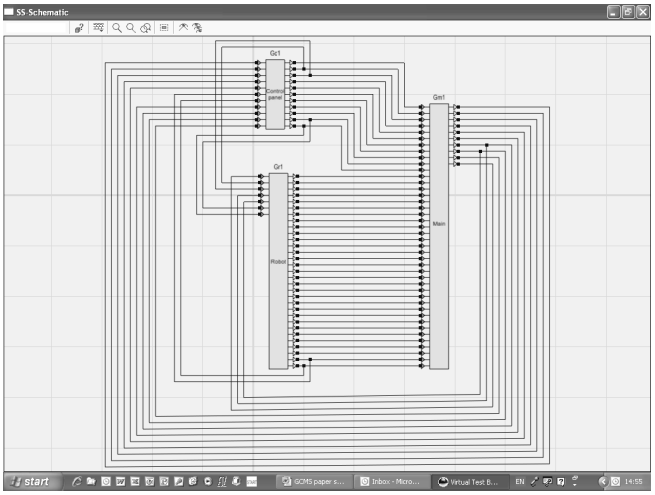

Figure 7 VTB Pro Genghis schematic

5. VISUALIZATION

Three-dimensional visualizations are realized from a VTB Pro simulation through a companion software package, Eye-Sys [IDV Inc]. A 3D model of the Genghis robot was developed in using Eye-Sys and this was linked to the VTB Pro simulation. The simulation calculates the global position and orientation of the robot body and orientation of each of the six legs relative to the body. This

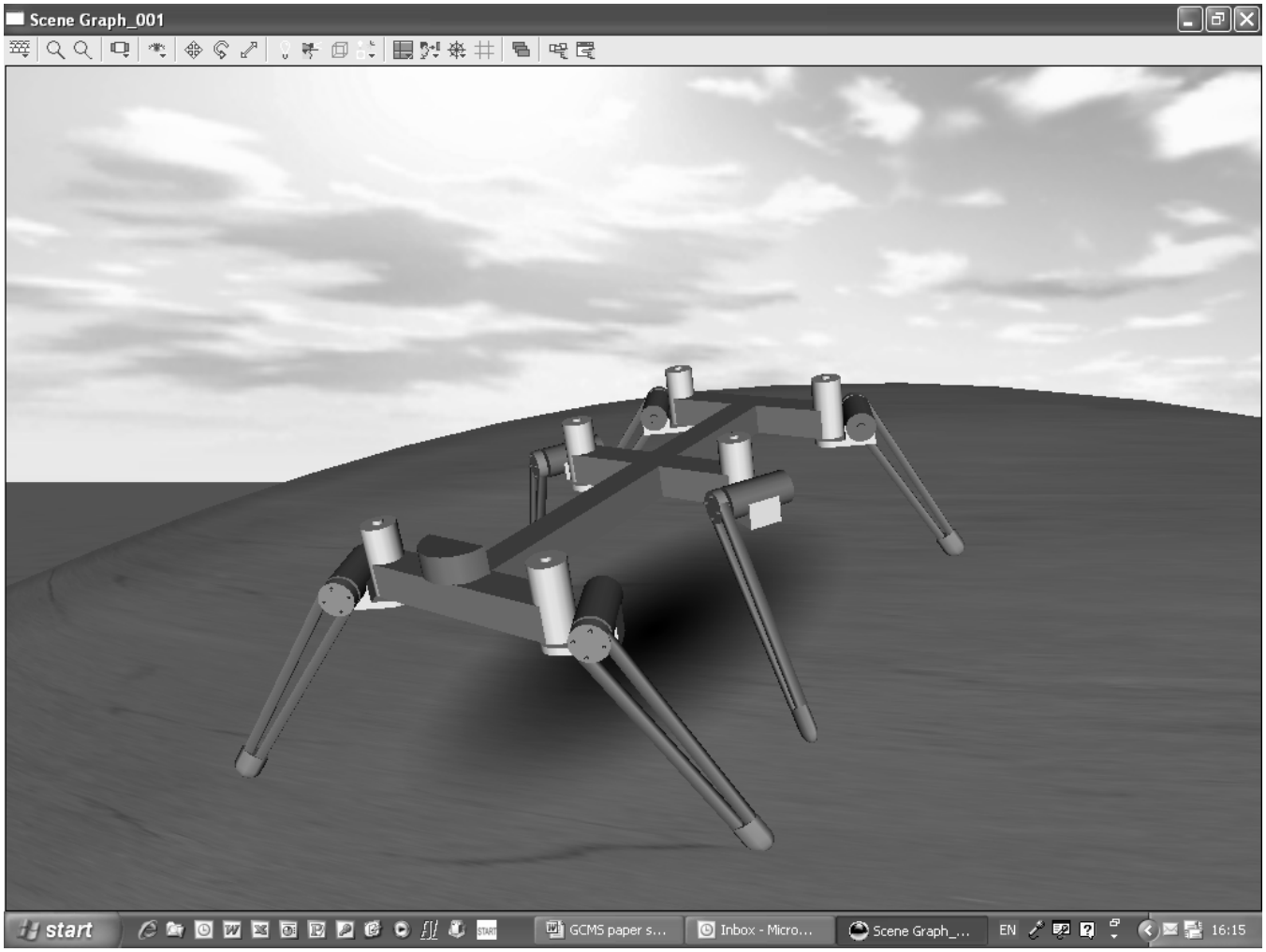

Figure 8 Genghis walks in VTB Pro

information is used to drive the simulation. Both the simulation and the visualization access an uneven terrain model over which the robot is constrained to walk. The visualization is complemented with ocean and sky models provided by the Eye-Sys software. Figure 8 shows Genghis striding out in its virtual environment.

6. DISCUSSION

The simulation, although a purely kinematic model, includes a considerable quantity of matrix calculations to determine the position and orientation of the robot components relative to the arbitrarily specified uneven terrain. Extensive use is also made of ESL's discontinuity detection mechanism to determine the precise times at which each of the robot's legs makes contact with the ground. Using a step size of 0.001*s*, the simulation runs approximately four times faster than real-time on a 3*GHz* Pentium processor.

Further development of the ESL – VTB integration would include improvement of the creation of .NET assemblies from ESL models, eliminating the need for textual editing of the source code to add package structures. Ideally there would be an option to generate .NET assemblies directly from a diagrammatically defined representation of the model in the Interactive Simulation Environment (ISE).

Another development would be the provision to create *natural ports* (i.e. a single port communicating both an across and through variable, such as voltage and current or velocity and force) in addition to *signal* ports (i.e. a single port communicating a single variable) in ESL, since these are dealt with very effectively in the VTB.

7. CONCLUSIONS

The latest stage of the development of a method of integrating models created using the ESL simulation tool

with the VTB has been described. The method involves generating .NET CLR assemblies from ESL embedded segment structures from which VTB entities are created and added to the VTB component database using a bespoke importer utility. The technique has been illustrated through an example of a multi segment simulation of an autonomous walking robot – Genghis. The concepts of importing external models using automatically generated wrapper code apply equally to other commonly used tools. Overall the exercise has demonstrated the versatility of the VTB to incorporate models developed in using other tools and the ease of linking visualization to a simulation.

Acknowledgements

This research has been supported by the Office of Naval Research through Awards N00014-04-1-0373 and N00014-08-1-0697. The author also acknowledges the contributions of Blake Langland and Claude Broughton of the University of South Carolina for the development of the ESL Importer and valuable suggestions.

References

Crosbie, R.E., Hay, J.L. and Pearce, J.G. 1981. "Simulation Studies with Modern Computer Structures". Final Report, (ESTEC Contract 4155/79), ESTEC, Noordwijk, The Netherlands.

Hay, J.L., Pearce, J.G., Crosbie, R.E. and Pallett, S. 1994. “ESL Simulation Tool”. Final Report, (ESTEC Contract 10011/92/NL/JG Work Order No. 2), ESTEC, Noordwijk, The Netherlands.

Pearce, J.G. 1993. "Six legged walking robot simulation". In proceedings of the United Kingdom Simulation Society Conference, (Keswick UK, 1993), UKSC 1-5.

Pearce, J.G. 2007. “Interfacing the ESL Simulation Language to the Virtual Test Bed”. In Proceedings of the 2007 Western Multiconference on Computer Simulation, (San Diego, CA, Jan 14-17). SCS, San Diego, CA, 166-171.

Pearce, J.G. and Crosbie, R.E. 2000. "ESL-ISE - A Simulation Tool Developed for the Space Industry". In Proceedings of the 2000 International Conference on Simulation and Multimedia in Engineering Education, (San Diego, CA, Jan 23-27). SCS, San Diego, CA, 115-120.

ISIM International Simulation Limited
http://isimsimulation.com

Interactive Data Visualizations Inc
http://www.idvinc.com